The OBSERVER: An Intelligent and Automated Video Surveillance System

Duarte Duque, Henrique Santos, and Paulo Cortez

University of Minho, Department of Information Systems, Campus de Azurém,
4800-058 Guimarães, Portugal
{duarteduque, hsantos, pcortez}@dsi.uminho.pt

Abstract. In this work we present a new approach to learn, detect and predict unusual and abnormal behaviors of people, groups and vehicles in real-time. The proposed OBSERVER video surveillance system acquires images from a stationary color video camera and applies state-of-the-art algorithms to segment and track moving objects. The segmentation is based in a background subtraction algorithm with cast shadows, highlights and ghost's detection and removal. To robustly track objects in the scene, a technique based on appearance models was used. The OBSERVER is capable of identifying three types of behaviors (normal, unusual and abnormal actions). This achievement was possible due to the novel N-ary tree classifier, which was successfully tested on synthetic data.

Keywords: Moving object detection, Tracking, Behavior detection.

1 Introduction

In the 70's, Tickner and Poulton, two researchers from the psychology field published a study [1] about the efficacy of human surveillance when dealing with a large number of cameras. The study has demonstrated that the level of attention and the accuracy of abnormal event detection decreases along the time. The authors also verified that human factors, such age and sex, can influence the reliability of incident detection.

A recent work [2] from the Department of Experimental Psychology, at the University of Bristol, was dedicated to study if potentially antisocial or criminal behavior could be predicted by humans when viewing real images recorded in CCTV. The outcome of a signal-detection analysis established that the estimates of the perceived magnitude of the evidence in favor of an incident were very similar for surveillance experts and novices, with an average true positive detection rate of 81%. The study suggests that prediction of future events is indeed possible, although it is imperfect. Moreover, the authors pointed out to the possibility of developing automatic units to detect abnormal events.

Some attempts to automatically detect and predict abnormal behaviors have been already presented. The ADVISOR system [3], aiming to detect vandalism acts, crowding situations and street fights, was one of the most relevant works in this field. It made use of a 3D model of a monitored area, where abnormal actions were previously defined by security experts, who described those acts using a predefined description language. This sort of approach led to a context dependent detection system,

A. Campilho and M. Kamel (Eds.): ICIAR 2006, LNCS 4141, pp. 898 – 909, 2006.

where all the objects from the scene and relations between those objects need to be defined.

Alessandro Mecocci in [4] introduces an architecture for an automatic real-time video surveillance system, capable of autonomously detecting anomalous behavioral events. The proposed system automatically adapts to different scenarios without any human intervention, and uses self-learning techniques to learn the typical behavior of the targets in each specific environment. Anomalous behaviors are detected if the observed trajectories deviate from the typical learned prototypes. Despite the significant contribution of Mecocci's work, it does not accomplish the particular features of a surveillance system, where typically the monitored area comprises different levels of security, i.e. the existence of restricted and public areas. Also, Mecocci made use of only spatial information, which is a significant lack of attributes for describing actions.

To overcome the identified problems, a project called OBSERVER, which is described in this paper, was started in 2004, aiming to automatically learn, detect and predict abnormal behaviors in public areas.

The paper is organized as follows. In section 2, the segmentation process, based on an adaptive background subtraction algorithm, which includes shadow, highlight and ghost detection techniques, is described. In section 3, we present the tracking algorithm and describe the appearance model approach. Then, in section 4, the main features of the automatic dataset generation are explained. The proposed method to predict abnormal behaviors and the creation of the N-ary tree classifier are also described. Finally, experimental results are presented in section 5 and, in section 6, we draw some conclusions.

2 The Segmentation Process

Segmenting moving objects is the first process of a computer based video surveillance system. It should provide accurate segmentation results in real-time. A good segmentation process should handle image source errors, such as white noise from the camera, cast shadows and highlights. To accomplish that, the OBSERVER system uses an adaptive background subtraction algorithm, combined with shadow, highlight and ghost detection and removal methods.

In the first step of the proposed segmentation process, given an image I and a background image B, both in the RGB color space, we select the moving pixels based on a fixed threshold. As outcome, a binary mask, called Primary Motion Mask (*PMM*) is generated.

$$PMM_n(x,y) = \begin{cases} 1 & if \quad \arg\max_{R,G,B} \left| I_n^{R,G,B}(x,y) - B_n^{R,G,B}(x,y) \right| > T \\ 0 & otherwise \end{cases} \tag{1}$$

This primary mask may be affected by error sources as we can see in Figure 1 (c). Therefore, the process continues with the detection of shadows and highlights from the *PMM*. This is achieved by a color space transformation of the acquired image from RGB to HSV, from which the shadows and highlight binary masks are produced, by the following operations:

$$SM_n(x,y) = \begin{cases} 1 & if \quad \alpha \le \dfrac{I_n^V(x,y)}{B_n^V(x,y)} \le \beta \quad \wedge \quad \left| I_n^S(x,y) - B_n^S(x,y) \right| \le \tau_S \quad \wedge \quad \left| I_n^H(x,y) - B_n^H(x,y) \right| \le \tau_H \\ 0 & otherwise \end{cases} \tag{2}$$

$$HM_n(x,y) = \begin{cases} 1 & if \quad \dfrac{1}{\beta} \le \dfrac{I_n^V(x,y)}{B_n^V(x,y)} \le \dfrac{1}{\alpha} \quad \wedge \quad \left| I_n^S(x,y) - B_n^S(x,y) \right| \le \tau_S \quad \wedge \quad \left| I_n^H(x,y) - B_n^H(x,y) \right| \le \tau_H \\ 0 & otherwise \end{cases} \tag{3}$$

The $I_n^H(x,y)$, $I_n^S(x,y)$ and $I_n^V(x,y)$ represent respectively the hue, saturation and value components of a pixel at coordinate *(x,y)* from the input image *I* at time *n*. The same notation was applied to the background image *B*.

In the above expressions, $0.65 \le \alpha \le 0.80$, $0.90 \le \beta \le 0.95$, $\tau S \le 15\%$ of digitizer saturation range and $\tau H \le 60$ degrees of color. This is explained in detail in previous work [5]. The *PMM* is then subtracted by *SM* and *HM*, generating a Motion Mask (*MM*).

$$MM_n = PMM_n \cap \neg SM_n \cap \neg HM_n \tag{4}$$

Concerning noise removal, the use of separated masks for shadow and highlight detection allows a better result than was obtained with a common mask. The noise is removed from each mask by a morphologic open operator followed by a close, using a 3x3 structuring element.

The resulting foreground regions from the *MM* are then filled in, in order to eliminate holes inside segmented blobs. The resulting blobs are therefore filtered by an area threshold, where small regions are considered as noise and removed from the *MM*. Using this mask, the blobs are labeled and grouped into objects to be tracked, as described in the next section.

Some of the detected objects may be ghosts, i.e. false positives originated by displacements of background objects. In those cases, ghosts regions need to be detected in the *MM*, in order to adapt the background model accordingly.

As stated before, an object can be composed by several segmented blobs.

$$Object_j = \bigcup_{i=0}^{N} Blob_i \qquad , N \in IN \tag{5}$$

We identify a segmented object as a ghost, only if all the blobs that constitute the object are classified as ghosts. This classification is determined as follows:

$$Blob_i = \begin{cases} ghost & if \quad \dfrac{Perimeter(\phi(I_n) \cap \phi(Blob_i))}{Perimeter(\phi(Blob_i))} < 10\% \\ motion & otherwise \end{cases} \tag{6}$$

where Φ denotes the edge detection operator.

The choice of the technique to be applied in the edge detection is critical. It must present good results and low response times. For this purpose the Canny, Sobel and Perwitt algorithms were tested.

Despite the result quality, the computational requirements for those algorithms exceeded significantly the time requirements for our application. To overcome this drawback, a faster and simple, yet efficient edge detection algorithm was used. It compares each pixel value with is four connected neighbor pixels. If the absolute

difference between a pixel and one of its neighbors is higher than a given threshold, that pixel is marked as an edge. Based on this description, the following operator will then produce a binary image with all the edges detected.

$$\Phi(D(x,y)) = \begin{cases} 1 \quad if \quad |D(x,y) - D(x-1,y-1)| > T \quad \vee \quad |D(x,y) - D(x-1,y+1)| > T \quad \vee \\ \qquad\quad |D(x,y) - D(x+1,y-1)| > T \quad \vee \quad |D(x,y) - D(x+1,y+1)| > T \\ 0 \quad otherwise \end{cases} \tag{7}$$

Finally, the background image is updated in the RGB and HSV color spaces. Pixels belonging to shadows, highlights and objects remain unchanged. Background pixels of segmented regions classifieds as ghosts are set with the actual image value. The remaining pixels are updated by an infinite impulse response filter in order to adapt the reference image to slight luminosity changes. The background updating can therefore be expressed by:

$$B_{n+1}(x,y) = \begin{cases} B_n(x,y) \quad if \quad motion \vee shadow \vee highlight \\ I_n(x,y) \quad if \quad ghost \\ \alpha \cdot B_n(x,y) + (1-\alpha) \cdot I_n(x,y) \quad otherwise \end{cases} \tag{8}$$

where α controls how fast are the changes introduced.

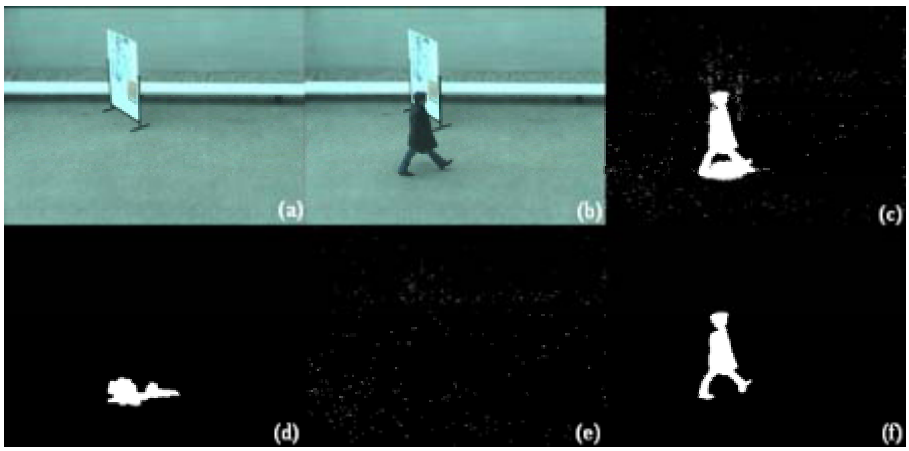

Fig. 1. (a) Background Image; (b) Current Image; (c) Primary Motion Mask; (d) Shadow Mask; (e) Highlight Mask; (f) Filtered Motion Mask

3 Tracking of Moving Objects

After segmenting moving objects, it is necessary to track them over the time, as long as they remain on the scene. For this task, a method based on appearance models was chosen.

Appearance models can be seen as dynamic memory representations of objects that are updated, at each frame, during the track operation. They are composed by two masks: the Appearance Image (*AI*), which maintains a color model of the object; and

the Probability Mask (*PM*), that represents the most likely shape that the object can assume. An example of an appearance model can be seen in Figure 2.

At each frame we will have a list of objects and, if in the presence of tracks, a list of tracks, respectively:

$O = \{O_1, O_2, ..., O_J\}$ with, $O_j = \{BB, M\}$

$T = \{T_1, T_2, ..., T_K\}$ with, $T_k = \{ID, Type, BB, AI, PM\}$

Where *BB* is the bounding box of the object, *M* is the object mask, *ID* is the track identifier, and *Type* indicates the object's class, i.e. "person", "group" or "vehicle". The tracking algorithm is divided in four steps: track setup; track alignment; pixel assignment; and track update.

3.1 Track Setup

In the track setup, a correspondence matrix *C*, which associates each object O_j to a track T_k, is created. This matrix with *JxK* dimension, where *J* is the number of segmented objects and *K* the number of detected tracks, is a binary matrix with value one in an element $C_{i,j}$, when the object O_j is associated with a track T_k.

This association is established if the currently segmented object O_j is overlapped by a previous track T_k. An example of a correspondence matrix is:

$$C_{j,k} = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

In this example, there are three objects and four tracks. The first segmented object (first line) is a merge of the first and second tracks, i.e. first and second columns, respectively. Also, the first and second objects derived from a split from the second track. The third object is single tracked by track three, and the track four was not found at the current frame.

If an object can't be assigned to at least one track, then a new track must be created for that object. In that case, the *AI* of the track is initialized with the color image of the segmented object, the *PM* is initialized with an intermediate value, and a new *ID* is assigned to the track. The correspondence matrix *C* is then processed, and objects that are associated to a same track are joined together, forming a macro-object (*MO*).

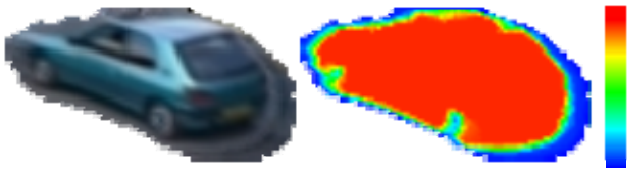

Fig. 2. Example of an Appearance Model. At left the appearance image, and at right the Probability Mask with scale.

3.2 Track Alignment

After performing the track setup, a list of *MO* and a list of tracks (T_k) are presented. In order to align the tracks, for each observed *MO* we will try to find the track displacement that best fits the appearance model of T_k with the *MO*. This is done for all the tracks assigned to a *MO*, and starting from those tracks whose objects are closest to the camera. The displacement (δ) is obtained by maximizing the fitting function P_{FIT}:

$$\delta = (\delta x, \delta y) = \arg\max_{\delta}\left(P_{FIT}\left(MO, T_k, \delta\right)\right) \tag{8}$$

where,

$$P_{FIT}\left(MO, T_k, \delta\right) = \frac{\sum_{x,y \in MO} P_{APP}\left(I^{R,G,B}(x,y), AI_k(x-\delta x, y-\delta y)\right) \cdot PM_k(x-\delta x, y-\delta y)}{\sum_{x,y \in T_k} PM_k(x,y)} \tag{9}$$

Considering that color components are uncorrelated and have equal variance, then:

$$P_{APP}\left(I^{R,G,B}, \mu^{R,G,B}\right) = \frac{1}{\sqrt{(2\pi)^3 \cdot \sigma^3}} \cdot e^{-\frac{1}{2}\cdot\left[\left(\frac{I^R-\mu^R}{\sigma}\right)^2+\left(\frac{I^G-\mu^G}{\sigma}\right)^2+\left(\frac{I^B-\mu^B}{\sigma}\right)^2\right]} \tag{10}$$

The displacement is initialized with the values of the previous track and continues the search by a gradient descent approach. When the displacement that maximizes the fitting function is achieved, the pixels of the macro-object that match to the track are excluded for the following tracks fitting.

3.3 Pixel Assignment

The pixel assignment is used to decide when a track is finished or not. Once the tracks are aligned, each pixel of a *MO* with a *PM* value greater than zero is assigned to one of the tracks that compete for that pixel. If the ratio between the number of assigned pixels and the number of probability mask pixels with value different than zero is lower than a predefined threshold, the track is removed.

The association of each pixel to a track is made based on the product of the pixel probability to belong to the appearance image by the pixel probability to belong to the track, which is $(x,y)\ \epsilon\ T_k$ for:

$$\arg\max_{k}\left(P_{APP}\left(I^{R,G,B}(x,y), AI_k(x-\delta x, y-\delta y)\right) \cdot PM_k(x-\delta x, y-\delta y)\right) \tag{11}$$

A *MO* pixel can be in one of three states. If it is assigned to the track T_k then we say it belongs to the set of the assigned pixels (AP_k). If the pixel doesn't have a value in the probability mask, then it is labeled as new. Otherwise, it is classified as missed pixel.

3.4 Track Update

The final step in the tracking procedure is the track update. In this step, the appearance image (*AI*) and the probability mask (*PM*) of single tracked objects are updated. Merged tracks will update only the position of the actual bounding box.

The appearance models are updated by the following rules:

$$PM_k^{t+1} = \begin{cases} \alpha \cdot PM_k^t(x,y) + (1-\alpha) & if \quad (x,y) \in AP_k \\ 0.5 & if \quad new \\ \alpha \cdot PM_k^t(x,y) & otherwise \end{cases} \tag{12}$$

$$AI_k^{t+1} = \begin{cases} \alpha \cdot AI_k^t(x,y) + (1-\alpha) \cdot I(x,y) & if \quad (x,y) \in AP_k \\ I(x,y) & if \quad new \\ AI_k^t(x,y) & otherwise \end{cases} \tag{13}$$

where, $0 \leq \alpha \leq 1$.

The appearance model of a tracked object remains in memory while the object is in the scene. After an object leaves the monitored area, the appearance model is deleted from the computer memory. As effect, if an object reenters in the scene, it will be detected as a new object, and not an extension of the previously observed movement. To overcome this issue, an approach that maintains, for a predefined period of time, the appearance models of departed objects, can be used. However, the RAM memory becomes a critical resource when dealing with a huge number of objects.

4 Behavior Detection and Prediction

The final goal of the OBSERVER system is to predict abnormal behaviors. For this propose the monitored temporal sequences of object attributes (object position, area, perimeter, velocity, etc) must be processed by a classifier in order to infer the type of action performed by an object. Such classifier will need to learn, from a dataset, the typical patterns of normal and abnormal events.

4.1 Automatic Dataset Generation

Typically, in an area under surveillance, the focus of attention falls into two types of events: unusual behaviors (e.g. a person running in a hotel lobby, or a car in wrong-way driving in a motorway) and violation of a restricted area (for example with pedestrians crossing a highway). The OBSERVER system was developed to successfully handle both types of events.

While the first type of events can be solved entirely by a classification algorithm, the violation of restricted areas requires human intervention in the system configuration stage. To accomplish that, the user only needs to mark the scene's restricted areas.

After the definition of the restricted areas, if there is any, the system starts to track objects moving in the scene. The observed tracks, along with the attributes of the objects, are recorded in real-time into a database. Tracks that overlap restricted areas are flagged as alarm events, the others are signaled as normal events. When the number of recorded tracks reaches to a predefined value, considered sufficient to the learning phase, then the classifier engine starts.

4.2 Constructing the N-Ary Tree Classifier

Before explaining the process of building the N-ary tree classifier, used to classify object behaviors, we will start by clarifying the idea behind the proposed classification system and the architecture of the classifier.

The behavior of an object can be described by a set of actions that it performs in a certain environment. From the security point of view we could expect, at least, three kinds of observations: *normal*, *unusual* or *abnormal* actions. Normal actions are those which are frequently observed and do not origin the violation of any restricted area. Unusual actions are those that are not common or have never occurred. When an action leads to a violation of a restricted area, then it should be classified as an abnormal action.

Besides this basic function, the OBSERVER will be able to predict those events from the registered object path. This takes us to a classifier that should respond to the following question: if an object, with specific properties, travels until a certain point, what is its probability to follow a path that will cause an abnormal event?

We propose such a classifier, using an N-ary tree, whose nodes are multivariable Gaussian distributions $N(\mu, \Sigma)$ in the object attribute's hyperplane, and have an abnormal probability (P_{ab}). To simplify the computation, we assume that the variances, in the covariance matrix Σ, are not correlated.

Has we can see in Figure 3, the classifier's tree is organized by layers. Each layer defines a period of time, so every track should be described by a sequence of nodes, each one from a different layer, starting in "layer 1", when the observed object enters the scene. The nodes present clusters of related points, i.e. Gaussian distributions, and subsequent layers are separated by a fixed time slice. For example, we can define that each layer will have a time period of ten frames.

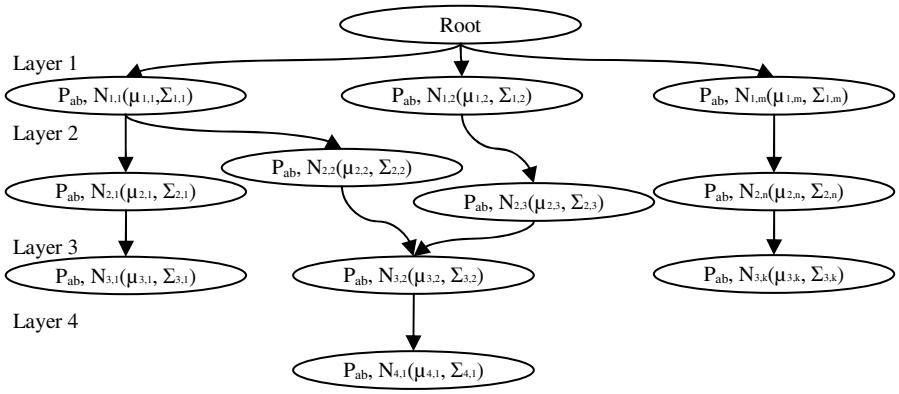

Fig. 3. Example of an N-ary tree classifier

To build the classifier, we use as input the set of prerecorded tracks. Each track is defined by a sequence of attribute vectors, describing the properties of an object at each sample. The following object attributes are considered: the 2D coordinates center-of-gravity; velocity; area; type descriptor (i.e. human, group or vehicle); and an alarm flag.

The first step consists on the partitioning of the data into equal time slices, i.e. layers. Then, for each layer of a track and for each attribute, the sampled data is processed in order to compute the mean value of the attribute for that layer. When this process finishes, the clustering is performed.

Since there is no information about the number of clusters in a layer, a clustering algorithm capable to infer the number of expected distributions should be used. For this

propose, an Expectation-Maximization algorithm with automatic cluster number detection based in k-fold cross-validation [6] was implemented, with k = 10, as described next.

4.3 Clustering the Data

Consider X a d-component column vector of object attributes, μ the d-component mean vector, Σ the d-by-d covariance matrix, and $|\Sigma|$ and Σ^{-1} its determinant and inverse, respectively.

For each layer there are N track samples, and initially the number of classes (C) is set to one. The first step of the k-fold cross-validation is to divide the dataset into ten fractions. Next, nine fractions are selected to compute the expectation and maximization steps. The remaining fraction is used to compute the log-likelihood. This procedure is executed ten times, in order to compute the log-likelihoods of all the fractions.

The final log-likelihood is calculated as the mean of the ten log-likelihoods. The value of classes (C) will be incremented while the log-likelihood (L) is not stabilized. After setting the number of clusters (C), the computation of the normal distributions is performed by the Expectation-Maximization algorithm, as described bellow.

Starting conditions:

$$\textit{foreach} \quad j \in C \quad \textit{do}$$

$$P(W_j) = 1/C$$

$$\mu_j = random(X)$$

$$\overline{\mu}_j = \sum_{i=1}^{N} X_i \Big/ N$$

$$\Sigma_j = \sum_{i=1}^{N} (X_i - \overline{\mu}_j)^2 \Big/ N$$

Expectation step:

$$P(X) = \sum_{j=1}^{K} \left[P(X \mid W_j) \cdot P(W_j) \right]$$

$$\textit{foreach} \quad j \in C \quad \textit{do}$$

$$P(W_j \mid X) = \frac{P(X \mid W_j) \cdot P(W_j)}{P(X)}$$

where,

$$P(X \mid W_j) = \frac{1}{(2\pi)^{d/2} \cdot |\Sigma_j|^{1/2}} \cdot e^{\left(-\frac{(X-\mu_j)^T \cdot \Sigma_j^{-1} \cdot (X-\mu_j)}{2} \right)}$$

Maximization step:

$$\textit{foreach} \quad j \in C \quad \textit{do}$$

$$\hat{P}(W_j) = \frac{\sum_{i=1}^{N} P(W_j \mid X_i)}{N}$$

$$\hat{\mu}_j = \frac{\sum_{i=1}^{N} \left[Xi \cdot P(W_j \mid X_i) \right]}{N \cdot \hat{P}(W_j)}$$

$$\hat{\Sigma}_j = \frac{\sum_{i=1}^{N} \left[P(W_j \mid X_i) \cdot (X_i - \hat{\mu}_j) \cdot (X_i - \hat{\mu}_j)^T \right]}{N \cdot \hat{P}(W_j)}$$

Stopping criteria:

$$\textit{do}$$

$$Expectation\_Step(\,)$$

$$Maximization\_Step(\,)$$

$$L = \sum_{i=1}^{N} \ln(P(X_i))$$

$$\textit{while}(|L - L_{old}| < \varepsilon)$$

When the stopping condition is satisfied, the mean and variance values of the distributions of a layer are merged. After executing the clustering over the entire dataset,

a set of clusters for each layer is obtained. The next step consists on building the classification tree, defining links between clusters of different layers.

4.4 Linking the N-Ary Tree Classifier

To build the classification tree, it is necessary to go through all the track sequences of the dataset. For each track, we will try to find a path (sequences of clusters, each cluster in a different layer) that represents it. When a dissimilar path is observed, a new branch is created. During this process of tree nodes linking, the information about the event type (i.e. normal or abnormal event) of each track and at each time slice is recorded within the clusters. At the end, all nodes of the tree will have the information about: mean and variance value of the multivariable normal distribution; and the number of normal and abnormal events observed by the cluster. In this way, it is possible to infer the probability of an object to exhibit an abnormal behavior, associated with a certain path.

5 Experimental Results

In this section we present the preliminary experiments to evaluate the proposed N-ary tree classifier. The dataset used to evaluate the classifier was artificially generated. For this propose, a semiautomatic track generator software, designated OBSERVER-TG, was developed. This tool allows the user to generate tracks in a scene, with configurable Gaussian noise over the object position, velocity, area and perimeter.

A dataset of 180 tracks with different lengths was created, with a restricted area, as shown in Figure 4. The set of generated tracks represent six different paths. Ten of the tracks violated the restricted area, originating alarm events. All the abnormal tracks started at bottom left of the scene, cross the road and turn left into the protected area.

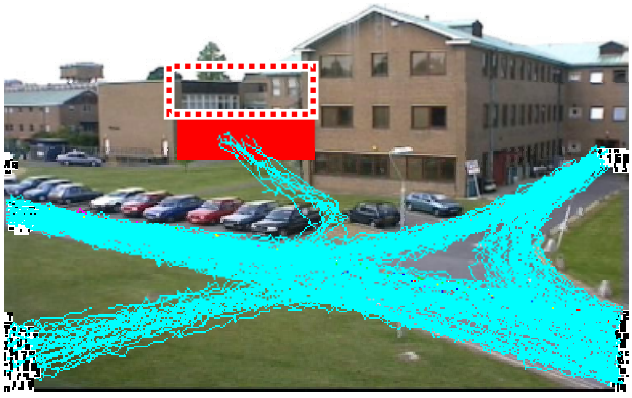

Fig. 4. Background overlapped by 180 tracks from the dataset and a restricted area (red box)

To evaluate the accuracy of the proposed classifier, the dataset was randomly fractioned in three parts. To access the system performance, a hold-out validation scheme [6] was adopted, where 2/3 of the simulated data was used for training (e.g. learning phase) and the remaining 1/3 for testing. Tracks that have probability of abnormal behavior greater than zero are classified as abnormal events, and sequences that have unobserved paths are classified as unusual events. The test results obtained in the three simulations (A, B, C) are presented in the table bellow.

Table 1. Test results of the behavior classifier

	Test Set		OBSERVER Test Results	
	NºTracks	Abnormal	Unsusual	Abnormal
A	60	5.00%	1.66%	3.33%
B	60	8.33%	5.00%	3.33%
C	60	3.33%	1.66%	1.66%

For instance, we can see a reasonable performance of the OBSERVER in the simulation A, with a test set that has 5% of abnormal tracks. The system has successful detected 3.33% of tracks as abnormal events, and 1.66% as unusual events. Summating the abnormal with the unusual results, we obtain the total of 5%.

When analyzing the results, we detected that all the normal tracks from the test dataset were correctly classified. This outcome is a consequence of the similarity of the generated tracks. In a real situation, when the type of observed paths is more chaotic, we can expect that a considerably amount of safety tracks will be classified as unusual events. In such situation, the system user should be prompted to classify unusual events as normal or abnormal behaviors, for use in future and refined classifiers.

6 Conclusions

In this work we presented a new approach to automatically detect and predict abnormal behaviors. The proposed solution is based on an adaptive background subtraction algorithm, an appearance model tracking technique and an N-ary tree classifier which has performed well with artificial test data. Moreover, the N-ary tree classifier possesses complementary features that can be exploited to find the most used paths and to analyze the occupancy of the monitored areas.

The classifier prototype was constructed as a stand alone application due to the expensive computation required by the clustering process and the necessary evaluation tasks. However, once finished, it seems to be sufficiently fast to meet the real-time constraints of a surveillance system.

Despite the confidence level on the results obtained, the validation of this solution still lacks of an extensive test with real data. Therefore, our main concern in the near future is to extend the experiments of the proposed classifier to different scenarios, with a greater number of paths, both from simulated and real-world data.

References

1. Tickner A.H., Poulton E.C.: Monitoring up to 16 synthetic television pictures showing a great deal of movement. Ergonomics, Vol. 16. (1973) 381-401
2. Troscianko T., Holmes A., Stillman J., Mirmehdi M., Wright D., Wilson A.: What happens next? The predictability of natural behaviour viewed through CCTV cameras. Perception, Vol. 33 (1). (2004) 87-101
3. INRIA, ADVISOR – Annotated Digital Video for Intelligent Surveillance and Optimised Retrieval, 2006, URL: http://www-sop.inria.fr/orion/ADVISOR/
4. Mecocci A., Pannozzo M., Fumarola A.: Automatic detection of anomalous behavioural events for advanced real-time video surveillance. International Symposium on Computational Intelligence for Measurement Systems and Applications. Lugamo (2003) 187-192
5. Duque D., Santos H., Cortez P.: Moving object detection unaffected by cast shadows, highlights and ghosts. IEEE International Conference on Image Processing. Genoa (2005) 413-416
6. Kohavi R.: A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence. San Francisco (1995) 1137-1143.